XCSL Tutorial

by *Marta Jacinto*
Giovani Rubert Librelotto
José Carlos Ramalho
Pedro Rangel Henriques

Abstract

XML brought the concept of well-formmedness to the world of structured documents. An XML document can be well-formed or valid. To be valid it just has to convey certain rules specified in a DTD or XML Schema. DTDs enable us to specify structure rules and a little bit of dynamic semantics. Additionally, XML Schemas enable us to specify some static semantics. However, there are applications where we need to specify more complex invariants or constraints.

Some of these constraints are structural but many of them are non-structural and have some degree of complexity. Here is where XCSL comes into scene, enabling to specify this extra semantics.

This tutorial presents an XML based architecture that enables the specification of constraints, the kind of constraints we can not specify with DTDs. XCSL is not just a language, it is also a processing model. We also discuss the general philosophy underlying the proposed approach, presenting the architecture of our semantic validation system, and we detail the respective processor.
To illustrate the use of the XCSL language and the subsequent processing, we present a complete case-study showing, step-by-step, the way we handled the various problems it raises. By making the analysis of this case-study and following the explanation, it should be easy to use XCSL with every family of documents one comes across.

Introduction

As a descendant of SGML, XML allows the specification of the documents' structure. This way, the documents will be valid syntactically. However, being sure that documents are correctly written from a syntactic point of view, does not assure the static semantics correctness. From the publisher's point of view, it is desirable not to produce invalid documents. Having this problem in mind, we have been working on a solution. That solution is based on the approach followed to specify programming languages: we use a specification language to define a set of contextual conditions over the textual content of elements and the attribute values that should be satisfied by an XML instance. Those conditions restrict the set of syntactically correct documents to the set of semantically valid ones. Concerning Content Constraining, we can classify constraints as belonging to one of the four categories defined below:

- Domain range checking:
  This is the most common constraint. We need this type of constraint when we want a certain content/value to be between a pair of values (inside a certain domain). Normally, this kind of constraint is used when data is of type date or numeric. We can specify this kind of constraint with XML Schema but not with DTDs. So, in this latter case we could use XCSL.

- Dependencies between two elements or attributes:
  We have cases where an attribute value depends on the value of another element or attribute located in a different branch of the document tree. These are clearly context dependent constraints. We can use neither XML Schema nor DTDs to specify these constraints. Having XCSL in these situations is fundamental.

- Pattern matching against a Regular Expression:
  Sometimes we need to guarantee that content follows a certain format (as in the case of dates: there are more than 100 formats, or telephone numbers). Neither DTDs nor XCSL have sopport for regular expressions. XML Schema should be used in these situations.

- Complex constraints:
  We group here all the remaining constraints and call them complex because they are usually weird: they require a nested loop behaviour or complex nested calculations. In this tutorial we have included some examples of this kind of constraints. These are clearly out of DTDs or XML Schemas scope.

This clear separation and the experience we gained working with XML documents allowed the consolidation of our approach (XCSL - XML Constraint Specification Language) to deal with the semantic specification problem.

XML Schemas and DTDs

XML Schema was created once the syntax of DTDs fell short of the requirements of the XML users. The aims of the W3C XML Schema Working Group were to create a language that would be more expressive than DTDs and written in XML Syntax. In addition, it would also allow authors to place restrictions on the elements' content and attribute values in terms of primitive datatypes found in most languages.

While using DTDs, to specify constraints, we need to use a constraining language (such as XCSL) and, consequently, need two documents to completely validate an XML instance. Constraining with XML-Schemas, on the other hand, means, for a particular set of constraints, using only one document to validate XML instances, instead of using both a DTD and a constraint document. Unfortunately, the range of constraints we can validate with XML-Schemas is far from the set we specified above.

To validate XML instances against a XML-Schema, several parsers are available. Those parsers are similar to the traditional XML validators; but now, instead of a DTD, they are driven by an XML-Schema.

XCSL

XCSL, XML Constraint Specification Language, is a domain specific language conceived with the purpose of allowing XML designers to restrict the content of XML documents. It is a simple, and small language useful to write contextual constraints over the textual value of XML elements and attributes.

Basically, the language provides the constructors necessary to define the actions to be taken by a semantic validator. Those actions will be triggered whenever a boolean expression, over the value of the attributes or the content of the elements, evaluates to false. It means that XCSL also designates a processor that traverses the document's internal representation (the tree) and checks its correctness from a semantic point of view (we shall stress that the structural correctness is validated by the parser that builds the tree).

Moreover, XCSL is an XML language; so it becomes possible to add restrictions to XML documents using an XML dialect. That approach offers document designers a complete XML framework to couple with syntax and semantics. The benefits of such an approach are obvious.

As shown in Figure 1, XCSL processing is very easy and fits in the processing of XML documents in a very natural way; that is, we can validate a document semantically the same way we validate it structurally: instead of the XML Parser (to validate the syntax), we invoke a XSL Transformer (to process its semantics). The XSL Stylesheet (which plays the role that the DTD plays in the syntactic validation) that drives the semantic processing, can be generated automatically from the XCSL specification.

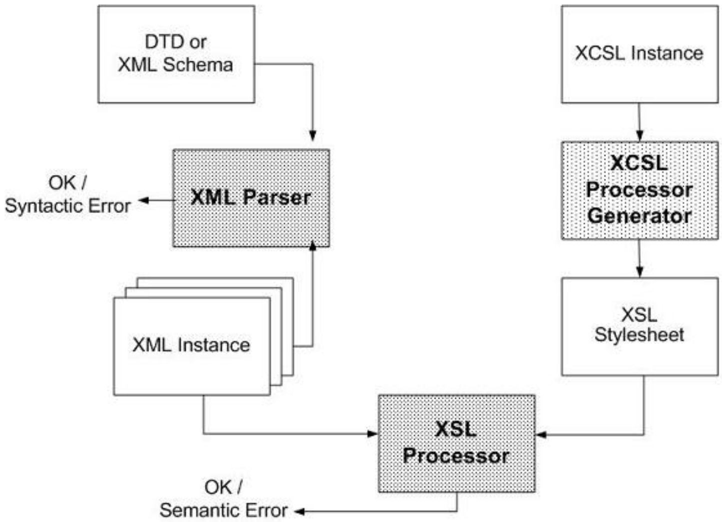

Figure 1: XCSL workflow

The XCSL Language

The first version of the Constraint Specification Language is formally defined in [Ramalho, 2000]. This exercise of formalization helped us to find the core structure of the language.

A specification in XCSL is composed by one or more tuples. Each tuple has three parts [Ramalho and Henriques, 1998]:

- Context Selector: As the name suggests, is the expression that selects the context where we want to enforce the constraint.
- Context Condition: The condition we want to enforce.
- Action: The action we want to trigger every time the condition does not hold.

In a more formal notation we can write:

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□

We could use a grammar to define the language, but as we stated before, we decided to use XSLT to specify the constraints; in order to be coherent, we needed an XML wrapper for the XSLT expressions (like in XSL). So, each XCSL specification is defined as an XML instance and the XCSL language is defined by a DTD (or an XML-Schema); the present version of the DTD that specifies XCSL (named xcsl.dtd) is shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- XCSL: XML Constraint Specification Language -->
<!ELEMENT CS (CONSTRAINT)+>
<!ATTLIST CS
    DTD CDATA #IMPLIED
    DATE CDATA #IMPLIED
    VERSION CDATA #IMPLIED>
<!ELEMENT CONSTRAINT (SELECTOR,LET*,CC,ACTION)>
<!ELEMENT SELECTOR EMPTY>
<!ATTLIST SELECTOR
    SELEXP CDATA #REQUIRED>
<!ELEMENT LET EMPTY>
<!ATTLIST LET
    NAME CDATA #REQUIRED
    VALUE CDATA #REQUIRED>
<!ELEMENT CC (#PCDATA|VARIABLE)*>
<!ELEMENT VARIABLE EMPTY>
<!ATTLIST VARIABLE
    SELEXP CDATA #REQUIRED>
```

```
<!ELEMENT ACTION (MESSAGE*)>
<!ELEMENT MESSAGE (#PCDATA|VALUE)*>
<!ELEMENT VALUE EMPTY>
<!ATTLIST VALUE
    SELEXP CDATA #REQUIRED>
```

Nowadays, XML-Schema has overcome DTDs as a definition of classes for XML instances according to what was said before. We also made that upgrade, however, as XML-Schema is much more verbose than the correspondent DTD, we decided to include here the DTD and just a diagrammatic description of the XCSL XML-Schema. That diagram is shown in Figure 2, as obtained with the XML Spy 4.1, from Altova.

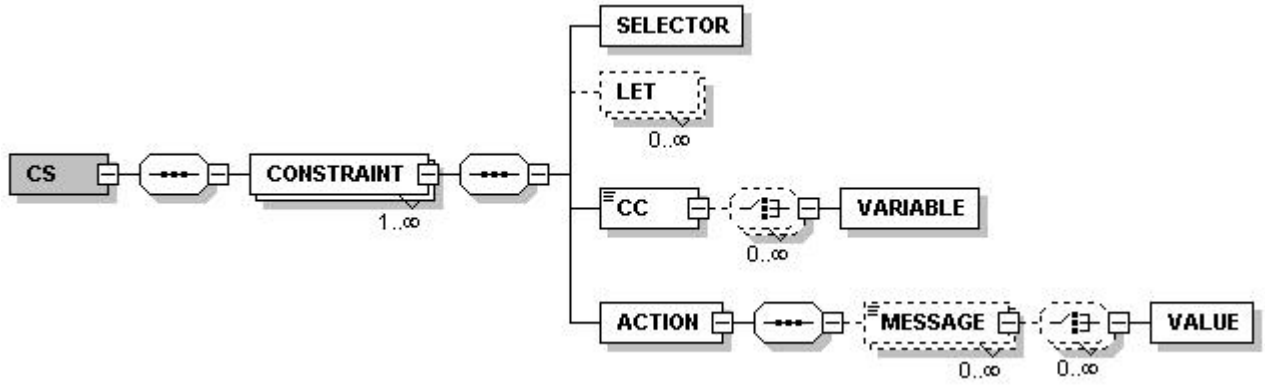

Figure 2: XCSL version 1.0 XML-Schema diagram

Elements

A XCSL constraint document consists of one or more CONSTRAINT elements. Each CONSTRAINT element specifies one constraint and the action that will be triggered when this constraint is not respected. The following table summarizes all the XCSL elements.

<table>
<tr><td>Element</td><td colspan="3">CS</td></tr>
<tr><td>Description</td><td colspan="3">Is the root element of an XML Constraint Specification Document. An XCSL constraint document consists of one or more CONSTRAINT elements.</td></tr>
<tr><td>Required</td><td colspan="3">true</td></tr>
<tr><td>Attributes</td><td colspan="3"><table>
<tr><td>DTD</td><td>name of DTD that is to be associated with the constraints being specified;</td><td>Required: false</td></tr>
<tr><td>DATE</td><td>XCSL document’s date;</td><td>Required: false</td></tr>
<tr><td>VERSION</td><td>XCSL document’s version.</td><td>Required: false</td></tr>
</table></td></tr>
<tr><td>Children</td><td colspan="3">CONSTRAINT</td></tr>
<tr><td>Parent</td><td colspan="3">/</td></tr>
<tr><td>DTD</td><td colspan="3"><pre>
<!ELEMENT CS (CONSTRAINT)+>
<!ATTLIST CS
      DTD CDATA #IMPLIED
      DATE CDATA #IMPLIED
      VERSION CDATA #IMPLIED>
</pre></td></tr>
</table>

Element	**CONSTRAINT**
Description	This element specifies a constraint and the action that will be triggered when this constraint is not respected. This element is formed by a sequence of elements: a SELECTOR element, zero or more LET elements, a CC element, and an ACTION element.
Required	true
Children	SELECTOR LET CC ACTION
Parent	CS
DTD	`<!ELEMENT CONSTRAINT (SELECTOR,LET*,CC,ACTION)>`

Element	**SELECTOR**		
Description	This element, as the name suggests, selects the context (the element or elements) within which the conditions should be tested.		
Required	true		
Attributes	SELEXP	this attribute holds an XPath (XML Path Language) expression that performs the selection.	Required: true
Parent	CONSTRAINT		
DTD	`<!ELEMENT SELECTOR EMPTY>`		

Element	**LET**		
Description	This element has an important role in complex constraints, it allows us to specify multi-step evaluations, enabling the simplification of very complex constraints. With this element we can save, in a variable (NAME) the result of an XPath expression applied to any XPath path, or still the set of values that belong to that context. It is an empty element.		
Required	false		
Attributes	NAME	it specifies the variable's name;	Required: true
	VALUE	holds an XPath expression applied to the context or to any XPath path.	Required: true
Parent	CONSTRAINT		

DTD	`<!ELEMENT LET EMPTY>`
	`<!ATTLIST LET`
	`NAME CDATA #REQUIRED`
	`VALUE CDATA #REQUIRED>`		

Element	**CC**
Description	It is an element that specifies the constraint that has to be verified – regular expression or XPath function. The action will be triggered whenever that expression or function is evaluated to false. It has a mixed content – text in which VARIABLE elements can occur any number of times.

Required	true
Children	VARIABLE
Parent	CONSTRAINT
DTD	`<!ELEMENT CC (#PCDATA|VARIABLE)*>`

Element	**VARIABLE**
Description	This element is used when we are enforcing a constraint over an element or attribute and we want to guarantee that it will be evaluated only if that element or attribute occurs in the document instance. If the element or attribute in question is absent from the document, the CONSTRAINT will not be evaluated.
Required	true
Attributes	SELEXP: it consists in the XPath path to that element or elements. Required: true
Parent	CC

DTD	`<!ELEMENT VARIABLE EMPTY>`
	`<!ATTLIST VARIABLE`
	`SELEXP CDATA #REQUIRED>`

Element	**ACTION**
Description	This is a required element that returns messages or results of evaluated expressions when the constraint being evaluated results in a false value. It consists of a sequence of messages. The contents of the MESSAGE element is returned when the ACTION element is unchained. It has mixed content: text with VALUE elements.
Required	true
Children	MESSAGE
Parent	CONSTRAINT
DTD	`<!ELEMENT ACTION (MESSAGE*)>`

Element	**MESSAGE**
Description	The content of this element is returned when the ACTION element is unchained. It has mixed content: text with VALUE element.
Required	false
Children	VALUE
Parent	ACTION
DTD	`<!ELEMENT MESSAGE (#PCDATA|VALUE)*>`

Element	**VALUE**

Description	This element is used to include some value of elements or attributes in the messages.
Required	false
Attributes	SELEXP: consists in a XPath path to the element or attribute which value we want to show. Required: true
Parent	MESSAGE

DTD	`<!ELEMENT VALUE EMPTY>`
	`<!ATTLIST VALUE`
	`SELEXP CDATA #REQUIRED>`

The XCSL Processor Generator

The XCSL processor generator is the main piece in our architecture. It takes an XML instance, written according to the XCSL language, and generates an XSL stylesheet that will test the specified constraints when processed by a standard XSL processor like Saxon or Xalan. The first versions of the generator were coded in Perl, using an XML down-translation module called XML::DT [Ramalho and Almeida, 1999]. The reason was that the task is quite complex and we needed an open tool with strong text processing capabilities. XML::DT is a perl module developed to process transformations over XML documents. It has some specific built-in operators and functions, but we can still use all the functionalities available in Perl. During the development of this generator we found some problems that had a strong impact in the final algorithm. The most important were:

- Optional or non-filled elements - suppose you have specified a constraint for every element named X; suppose that element X is optional and in certain parts of the instance the user did not insert it; the system will trigger the action for every nonexistent element X (in XSL the absence of an element that is part of a condition will make that condition always false).
- Ambiguity in context selection - until now, we have just said that an XCSL specification is composed by a set of constraints; we did not say that these constraints should be disjoint in terms of context; in some cases there is a certain overlap between the contexts of different conditions; this overlap will cause an error when transposed to XSL; XSL processors can only match one context at a time; there is one solution to this problem that is to run each constraint in a different mode (in XSL each mode corresponds to a different traversal of the document tree).

After some iterations and after solving those problems, the main algorithm looks like the following:

1. Convert each CONSTRAINT element into an xsl:template
2. Use attribute SELEXP for the xsl:template match attribute
3. Convert CC element into a predicate: [...], inside match attribute
4. Convert each "stamped" path (VARIABLE element) into a predicate
5. Convert each LET element into an xsl:variable
6. Put MESSAGE contents inside the template body converting:
    - each VALUE element into an xsl:value-of element
7. Filter all remaining text nodes

Recently, we have been developing the XSL version of the generator. The main function, which generates a template for each constraint, looks like the following:

```
<xsl:template match="CONSTRAINT">
    <xsl:variable name="cc">
        <xsl:apply-templates select="CC"/>
    </xsl:variable>
    <xsl:variable name="sel" select="SELECTOR/@SELEXP"/>
    <xsl:variable name="pred">
        <xsl:apply-templates select="CC/VARIABLE" mode="pred"/>
```

```
    <xsl:variable>
    <xsl:comment>
        ..................NEW CONSTRAINT...................
    </xsl:comment>
    <my:template mode="constraint{count(preceding-sibling::*)+1}"
    match="{$sel}{$pred}">
        <my:if test="not({$cc})">
            <err-message>
                <xsl:apply-templates select="ACTION"/>
            </err-message>
        </my:if>
    </my:template>
    <my:template match="text()" priority="-1"
    mode="constraint{count(preceding-sibling::*)+1}">
        <!-- strip characters -->
    </my:template>
</xsl:template>
```

Let us now go through a very simple example. Consider the following register:

```
<?xml version="1.0"?>
<cd>
    ...
    <price>32.00
    ...
</cd>
```

We want to ensure that every cd's price is within a certain range (for instance from 0 to 100). In order to do that, we specify the following constraint:

```
<?xml version="1.0"?>
<CS>
    <CONSTRAINT>
        <SELECTOR SELEXP="/cd/price"/>
        <CC>(.>0) and (.<100)
        <ACTION>
            <MESSAGE>Price out of range!
        </ACTION>
    </CONSTRAINT>
</CS>
```

This document, when processed by our generator, will generate the following XSL stylesheet:

```
<xsl:stylesheet version="1.0">
    <xsl:template match="/">
        <doc-status>
            <xsl:apply-templates mode="constraint1"/>
        </doc-status>
    </xsl:template>
    <!--..................NEW CONSTRAINT.....................-->
    <xsl:template mode="constraint1" match="/cd/price">
        <xsl:if test="not((.>0) and (.<100))">
            <err-message>Price out of range!
        </xsl:if>
    </xsl:template>
    <xsl:template match="text()" priority="-1" mode="constraint1"/>
    <xsl:template match="text()" priority="-1"/>
</xsl:stylesheet>
```

When applied to XML instances, this stylesheet will generate error messages whenever the constraint does not evaluate to true. Next we present a case-study. It begins with a brief introduction, followed by a DTD designed for this particular family of documents, the complete restrictions document and the stylesheet generated.

Afterwards, we analyse each constraint separately explaining the goal and the method used to solve each problem. Finally, we show a set of applications, one for each possible sub-kind of documents.

The Reservations Example

Let us suppose we want to use an XML document to register the reservations of a certain hotel. To do that, we design a DTD which allows one or more rooms to be reported. Each room will have two sub-elements: a group in charge (gicharge, made up of two employees - picharge) and a set of events; and two attributes: floor - to indicate the location of the room, and id - to uniquely identify the room. Each events element has any number of event elements, each one referring to an event for which the room is booked. Each event has three sub-elements: date - made up of the beginning date and the final date, companies - the set of organizer companies, title - the title of the event in which any number of compn (company name) and tradem (trade-mark) elements may occur. Each company is identified by a name (compn), an organizer and a contact (compc).

After validating the structure of the document, at least four semantic constraints must be validated in order to have a completely valid document.

These constraints are: first - the value of every floor attribute descendant of the room elements must be 12 or less (or any other number that represents the number of floors the hotel has); second - for each event, the final date must occur after the beginning date; third - every contact must have a valid format (only numbers, exactly nine, and begin either with a 2, 91, 93 or 96 - in the portuguese situation); fourth - every compn element that occurs inside a certain title element must be of an organizer company. Notice that these four constraints correspond to the four categories of constraints enumerated in the Introduction.

DTD:

```
<!ELEMENT reservations (room)+>
<!ELEMENT room (gicharge,events)>
<!ATTLIST room
    floor CDATA #REQUIRED
    id ID #REQUIRED>
<!ELEMENT gicharge (picharge,picharge)>
<!ELEMENT picharge (#PCDATA)>
<!ELEMENT events (event)*>
<!ELEMENT event (date,companies,title)>
<!ELEMENT date (dateb,datef)>
<!ELEMENT dateb (#PCDATA)>
<!ELEMENT datef (#PCDATA)>
<!ATTLIST dateb
    value CDATA #REQUIRED>
<!ATTLIST datef
    value CDATA #REQUIRED>
<!ELEMENT companies (company)+>
<!ELEMENT company (compn, organizer,compc)>
<!ELEMENT compn (#PCDATA)>
<!ELEMENT organizer (#PCDATA)>
<!ELEMENT compc (#PCDATA)>
<!ELEMENT title (#PCDATA|compn|tradem)*>
<!ELEMENT tradem (#PCDATA)>
```

Constraints:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<CS>
    <!-- 1 -->
    <CONSTRAINT>
        <SELECTOR SELEXP="/reservations/room"/>
        <CC>
        @floor <= 12
    </CC>
        <ACTION>
```

```
            <MESSAGE>
            The floor number <VALUE SELEXP="@floor"/> does not exist.
        </MESSAGE>
        </ACTION>
    </CONSTRAINT>
    <!-- 2 -->
    <CONSTRAINT>
        <SELECTOR SELEXP="//room/events/event/date"/>
        <CC>
        dateb/@value <= datef/@value
    </CC>
        <ACTION>
            <MESSAGE>
            The final date: <VALUE SELEXP="datef"/> occurs before the beginning
            date: <VALUE SELEXP="dateb"/> -this is not allowed.
        </MESSAGE>
        </ACTION>
    </CONSTRAINT>
    <!-- 3 -->
    <CONSTRAINT>
        <SELECTOR SELEXP="//compc"/>
        <CC>
        string-length(number(.)) = 9 and (substring(.,1,1)=2 or substring(.,1,2)=91 or
        substring(.,1,2)=93 or substring(.,1,2)=96)
    </CC>
        <ACTION>
            <MESSAGE>
            The contact for the company <VALUE SELEXP="../compn"/> is not a valid
            phone number.
        </MESSAGE>
        </ACTION>
    </CONSTRAINT>
    <!-- 4 -->
    <CONSTRAINT>
        <SELECTOR SELEXP="//title/compn"/>
        <LET NAME="keycompn" VALUE="."/>
        <CC>
    (count(../../companies/company[compn=$keycompn]) >= 1)
    </CC>
        <ACTION>
            <MESSAGE>
        The title of the event must not contain any company’s name outside the set of
        organizer companies, as <VALUE SELEXP="."/> in a reservation of the room
        <VALUE SELEXP="../../../../@id"/>.
      </MESSAGE>
        </ACTION>
    </CONSTRAINT>
</CS>
```

Stylesheet generated:

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?><!--
      Preprocessor for the XCSL Language
      http://www.di.uminho.pt/~jcr/PROJS/xcsl-www/
      Copyright (C) 2001 José Carlos Ramalho
      Permission to use granted under GPL or MPL.

      Version: 3.0
    -->
<my:stylesheet xmlns:my="http://www.w3.org/1999/XSL/Transform"
               xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
   <my:output method="xml" omit-xml-declaration="no" encoding="iso-8859-1"
    standalone="yes" indent="yes"/>
   <my:template match="/">
      <doc-status>
         <my:apply-templates mode="constraint1"/>
```

```
        <my:apply-templates mode="constraint2"/>
        <my:apply-templates mode="constraint3"/>
        <my:apply-templates mode="constraint4"/>
      </doc-status>
   </my:template><!--

  .......................NEW CONSTRAINT.......................
  -->
   <my:template mode="constraint1" match="/reservations/room">
      <my:if test="not(
        @floor <= 12
    )">
         <err-message>
            The floor number <my:value-of select="@floor"/> does not exist.
        </err-message>
      </my:if>
   </my:template>
   <my:template match="text()" priority="-1" mode="constraint1"/><!--

  .......................NEW CONSTRAINT.......................
  -->
   <my:template mode="constraint2" match="//room/events/event/date">
      <my:if test="not(
        dateb/@value <= datef/@value
    )">
         <err-message>
            The final date: <my:value-of select="datef"/> occurs before the beginning
            date: <my:value-of select="dateb"/> -this is not allowed.
        </err-message>
      </my:if>
   </my:template>
   <my:template match="text()" priority="-1" mode="constraint2"/><!--

  .......................NEW CONSTRAINT.......................
  -->
   <my:template mode="constraint3" match="//compc">
      <my:if test="not(
        string-length(number(.)) = 9 and (substring(.,1,1)=2 or substring(.,1,2)=91 or
        substring(.,1,2)=93 or substring(.,1,2)=96)
    )">
         <err-message>
            The contact for the company <my:value-of select="../compn"/> is not a
            valid phone number.
        </err-message>
      </my:if>
   </my:template>
   <my:template match="text()" priority="-1" mode="constraint3"/><!--

  .......................NEW CONSTRAINT.......................
  -->
   <my:template mode="constraint4" match="//title/compn">
      <my:variable name="keycompn">
         <my:value-of select="."/>
      </my:variable>
      <my:if test="not(
     (count(../../companies/company[compn=$keycompn]) >= 1)
     )">
         <err-message>
        The title of the event must not contain any company's name outside the set
        of organizer companies, as <my:value-of select="."/> in a reservation of the room
        <my:value-of select="../../../../@id"/>.
      </err-message>
      </my:if>
   </my:template>
   <my:template match="text()" priority="-1" mode="constraint4"/>
   <my:template match="text()" priority="-1"/>
</my:stylesheet>
```

The constraints one at a time:

The value of every floor attribute descendant of the room elements must be 12 or less.

Constraint 1:

```
<CONSTRAINT>
    <SELECTOR SELEXP="/reservations/room"/>
    <CC>
        @floor <= 12
    </CC>
    <ACTION>
        <MESSAGE>
        The floor number <VALUE SELEXP="@floor"/> does not exist.
    </MESSAGE>
    </ACTION>
</CONSTRAINT>
```

We check that the value of the floor attribute of each /reservations/room is less than or equal to 12. We set the context to /reservations/room analysing every occurence of this.

For each event, the final date must occur after the beginning date.

Constraint 2:

```
<CONSTRAINT>
    <SELECTOR SELEXP="//room/events/event/date"/>
    <CC>
        dateb/@value <= datef/@value
    </CC>
    <ACTION>
        <MESSAGE>
            The final date: <VALUE SELEXP="datef"/> occurs before the beginning date:
            <VALUE SELEXP="dateb"/> -this is not allowed.
        </MESSAGE>
    </ACTION>
</CONSTRAINT>
```

We set the context to //room/events/event/date, to analyse every occurence of date. Then we check if dateb is previous to datef.

Every contact must have a valid format - only numbers, in number of nine, and begin either with a 2, 91, 93 or 96.

Constraint 3:

```
<CONSTRAINT>
    <SELECTOR SELEXP="//compc"/>
    <CC>
        string-length(number(.)) = 9 and (substring(.,1,1)=2 or substring(.,1,2)=91 or
        substring(.,1,2)=93 or substring(.,1,2)=96)
    </CC>
    <ACTION>
        <MESSAGE>
            The contact for the company <VALUE SELEXP="../compn"/> is not a valid phone
            number.
        </MESSAGE>
    </ACTION>
</CONSTRAINT>
```

The context is now set to //compc. We verify that the contact is only made up of digits by checking that the length of the number obtained after the application of the number function -which returns only the digits, for instance number(232s)=232- is 9. Afterwards we check if the beginning substring of the contact is either equal to 2, 91, 93 or 96. Finally we make the interception of these two boolean values.

Every compn element that occurs inside a certain title element must be of an organizer company.

Constraint 4:

```
<CONSTRAINT>
    <SELECTOR SELEXP="//title/compn"/>
    <LET NAME="keycompn" VALUE="."/>
    <CC>
        (count(../../companies/company[compn=$keycompn]) >= 1)
    </CC>
    <ACTION>
         <MESSAGE>
            The title of the event must not contain any company's name outside
            the set of organizer companies, as <VALUE SELEXP="."/> in a reservation of th
            room <VALUE SELEXP="../../../../@id"/>.
         </MESSAGE>
    </ACTION>
</CONSTRAINT>
```

We use a LET element to place in keycompn the list of values that the compn element assumes in the //title/compn context. Later, the number of occurrences of each instance of compn (in companies/company descendant of the current's title event) in the keycompn list is counted and compared with 1. The action will be triggered whenever the negation of "greater than or equal" occurs, i.e., the name does not occur as an organizer of the event.

Applications:

Application 1:

Documents where the value of every floor attribute descendant of the room elements is 12 or less; for each event, the final date occurs after the beginning date; every contact has a valid format; and every compn element that occurs inside a certain title element refers an organizer company. The result was a document with no errors.

XML instance:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE reservations SYSTEM "reserv.dtd">
<reservations>
    <room floor="0" id="s1">
        <gicharge>
            <picharge>Ana</picharge>
            <picharge>Manuel</picharge>
        </gicharge>
        <events>
            <event>
                <date>
                    <dateb value="20010515">15th May 2001</dateb>
                    <datef value="20010515">15th May 2001</datef>
                </date>
                <companies>
                    <company>
                        <compn>Candle</compn>
                        <organizer>Gabriel</organizer>
                        <compc>214848737</compc>
                    </company>
                </companies>
                <title>
                    <compn>Candle</compn>Net</title>
            </event>
            <event>
                <date>
                    <dateb value="20010626">26th June 2001</dateb>
                    <datef value="20010626">26th June 2001</datef>
                </date>
                <companies>
                    <company>
```

```
                        <compn>EMC</compn>
                        <organizer>Flavio</organizer>
                        <compc>219458372</compc>
                    </company>
                </companies>
                <title>
                    <compn>EMC</compn> developments for <tradem>OS/390</tradem>
                </title>
            </event>
        </events>
    </room>
    <room floor="1" id="s2">
        <gicharge>
            <picharge>Ana</picharge>
            <picharge>Manuel</picharge>
        </gicharge>
        <events/>
    </room>
    <room floor="1" id="s3">
        <gicharge>
            <picharge>José</picharge>
            <picharge>Rita</picharge>
        </gicharge>
        <events>
            <event>
                <date>
                    <dateb value="20010524">24th May 2001</dateb>
                    <datef value="20010525">25th May 2001</datef>
                </date>
                <companies>
                    <company>
                        <compn>SOL-S</compn>
                        <organizer>Maria</organizer>
                        <compc>213487659</compc>
                    </company>
                    <company>
                        <compn>CheckPoint</compn>
                        <organizer>Carlos</organizer>
                        <compc>224357985</compc>
                    </company>
                    <company>
                        <compn>Remedy</compn>
                        <organizer>Bruno</organizer>
                        <compc>218705464</compc>
                    </company>
                </companies>
                <title>EBusiness2000 - <compn>CheckPoint</compn> and <compn>Remedy</compn
                </title>
            </event>
        </events>
    </room>
</reservations>
```

The generated file:

```
<?xml version="1.0" encoding="iso-8859-1" standalone="yes" ?>
<doc-status />
```

Application 2:

Documents where **the value of one or more floor attributes descendant of the room elements is greater than 12**; for each event, the final date occurs after the beginning date; every contact has a valid format; and every compn element that occurs inside a certain title element refers to an organizer company. For instance if we have two room elements and one is said to be located in the 14th floor, the result will be a document showing the error:

XML instance:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE reservations SYSTEM "reserv.dtd">
<reservations>
    <room floor="14" id="s1">
        <gicharge>
            <picharge>Ana</picharge>
            <picharge>Manuel</picharge>
        </gicharge>
        <events>
            <event>
                <date>
                    <dateb value="20010626">26th June 2001</dateb>
                    <datef value="20010626">26th June 2001</datef>
                </date>
                <companies>
                    <company>
                        <compn>EMC</compn>
                        <organizer>Flavio</organizer>
                        <compc>219458372</compc>
                    </company>
                </companies>
                <title>
                    <compn>EMC</compn> developments
                </title>
            </event>
        </events>
    </room>
    <room floor="1" id="s3">
        <gicharge>
            <picharge>José</picharge>
            <picharge>Bonifácio</picharge>
        </gicharge>
        <events>
            <event>
                <date>
                    <dateb value="20010524">24th May 2001</dateb>
                    <datef value="20010525">25th May 2001</datef>
                </date>
                <companies>
                    <company>
                        <compn>Remedy</compn>
                        <organizer>Bruno</organizer>
                        <compc>218705464</compc>
                    </company>
                </companies>
                <title>EBusiness2000 - <compn>Remedy</compn> products
                </title>
            </event>
        </events>
    </room>
</reservations>
```

The generated file:

```
<?xml version="1.0" encoding="iso-8859-1" standalone="yes"?>
<doc-status>
    <err-message>
            The floor number 14 does not exist.
        </err-message>
</doc-status>
```

Application 3:

Documents where the value of every floor attribute descendant of the room elements is 12 or less; **for one or more events, the final date occurs before the beginning date**; every contact has a valid format; and every compn element that occurs inside a certain title element refers an organizer company. For instance if we have two event elements and both have wrong dates, the result will be a document showing the errors.

XML instance:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE reservations SYSTEM "reserv.dtd">
<reservations>
    <room floor="0" id="s1">
        <gicharge>
            <picharge>Ana</picharge>
            <picharge>Manuel</picharge>
        </gicharge>
        <events>
            <event>
                <date>
                    <dateb value="20010515">15th May 2001</dateb>
                    <datef value="20010513">13th May 2001</datef>
                </date>
                <companies>
                    <company>
                        <compn>Promosoft</compn>
                        <organizer>Carlos</organizer>
                        <compc>219878586</compc>
                    </company>
                </companies>
                <title>
                    Portal/SME</title>
            </event>
            <event>
                <date>
                    <dateb value="20010626">26th June 2001</dateb>
                    <datef value="20010625">25th June 2001</datef>
                </date>
                <companies>
                    <company>
                        <compn>EMC</compn>
                        <organizer>Flavio</organizer>
                        <compc>219458372</compc>
                    </company>
                </companies>
                <title>
                    <tradem>OS/390</tradem>
                </title>
            </event>
        </events>
    </room>
</reservations>
```

The generated file:

```
<?xml version="1.0" encoding="iso-8859-1" standalone="yes"?>
<doc-status>
    <err-message>
            The final date: 13th May 2001 occurs before the beginning date:
            15th May 2001 -this is not allowed.
        </err-message>
    <err-message>
            The final date: 25th June 2001 occurs before the beginning date:
            26th June 2001 -this is not allowed.
        </err-message>
</doc-status>
```

Application 4:

Documents where the value of every floor attribute descendant of the room elements is 12 or less; for each event, the final date occurs after the beginning date; **one or more contacts have an invalid format**; and every compn element that occurs inside a certain title element refers an organizer company. For instance if we have three company elements and two of them have an invalid format, the result will be a document showing the errors.

XML instance:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE reservations SYSTEM "reserv.dtd">
<reservations>
    <room floor="1" id="s3">
        <gicharge>
            <picharge>José</picharge>
            <picharge>Bonifácio</picharge>
        </gicharge>
        <events>
            <event>
                <date>
                    <dateb value="20010524">24th May 2001</dateb>
                    <datef value="20010525">25th May 2001</datef>
                </date>
                <companies>
                    <company>
                        <compn>SOL-S</compn>
                        <organizer>Maria</organizer>
                        <compc>413487659</compc>
                    </company>
                    <company>
                        <compn>CheckPoint</compn>
                        <organizer>Carlos</organizer>
                        <compc>224357985</compc>
                    </company>
                    <company>
                        <compn>Remedy</compn>
                        <organizer>Bruno</organizer>
                        <compc>818705464</compc>
                    </company>
                </companies>
                <title>EBusiness2000 - <compn>CheckPoint</compn>
                and <compn>Remedy</compn>
                </title>
            </event>
        </events>
    </room>
</reservations>
```

The generated file:

```
<?xml version="1.0" encoding="iso-8859-1" standalone="yes"?>
<doc-status>
    <err-message>
            The contact for the company SOL-S is not a valid phone number.
        </err-message>
    <err-message>
            The contact for the company Remedy is not a valid phone number.
        </err-message>
</doc-status>
```

Application 5:

Documents where the value of every floor attribute descendant of the room elements is 12 or less; for each

event, the final date occurs after the beginning date; every contact has a valid format; and **one or more compn elements occuring inside a certain title element refer to a non-organizer company**. For instance if we have two events, in the first one's title the only compn that occurs refers to a non-organizer company, and in the second one's title the second and third compn that occur refer to non-organizer companies, the result will be a document showing the errors.

XML instance:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE reservations SYSTEM "reserv.dtd">
<reservations>
    <room floor="0" id="s1">
        <gicharge>
            <picharge>Ana</picharge>
            <picharge>Manuel</picharge>
        </gicharge>
        <events>
            <event>
                <date>
                    <dateb value="20010626">26th June 2001</dateb>
                    <datef value="20010626">26th June 2001</datef>
                </date>
                <companies>
                    <company>
                        <compn>EMC</compn>
                        <organizer>Flavio</organizer>
                        <compc>219458372</compc>
                    </company>
                </companies>
                <title>
                    <compn>EMC3</compn> developments
                </title>
            </event>
        </events>
    </room>
    <room floor="1" id="s3">
        <gicharge>
            <picharge>José</picharge>
            <picharge>Bonifácio</picharge>
        </gicharge>
        <events>
            <event>
                <date>
                    <dateb value="20010524">24th May 2001</dateb>
                    <datef value="20010525">25th May 2001</datef>
                </date>
                <companies>
                    <company>
                        <compn>SOL-S</compn>
                        <organizer>Coimbra</organizer>
                        <compc>213487659</compc>
                    </company>
                    <company>
                        <compn>CheckPoint</compn>
                        <organizer>Esteves</organizer>
                        <compc>224357985</compc>
                    </company>
                    <company>
                        <compn>Remedy</compn>
                        <organizer>Botas</organizer>
                        <compc>218705464</compc>
                    </company>
                </companies>
                <title>EBusiness2000 - <compn>CheckPoint</compn> and
                <compn>RemedyA</compn> and <compn>CA</compn>
                </title>
```

```
            </event>
        </events>
    </room>
</reservations>
```

The generated file:

```
<?xml version="1.0" encoding="iso-8859-1" standalone="yes"?>
<doc-status>
    <err-message>
        The title of the event must not contain any company's name outside
        the set of organizer companies, as EMC3 in a reservation of the room
        s1.
      </err-message>
    <err-message>
        The title of the event must not contain any company's name outside the
        set of organizer companies, as RemedyA in a reservation of the room
        s3.
      </err-message>
    <err-message>
        The title of the event must not contain any company's name outside the
        set of organizer companies, as CA in a reservation of the room
        s3.
      </err-message>
</doc-status>
```

Application 6:

Documents where a combination of the previous four situations occur, i.e, one or more conditions are violated. For instance if we have one event, the final date occurs before the beginning date, one contact does not have the allowed form and two of the compn elements which occur in the title are of non-organizer companies, the result will be a document showing the errors.

XML instance:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE reservations SYSTEM "reserv.dtd">
<reservations>
    <room floor="1" id="s3">
        <gicharge>
            <picharge>José</picharge>
            <picharge>Bonifácio</picharge>
        </gicharge>
        <events>
            <event>
                <date>
                    <dateb value="20010524">24th May 2001</dateb>
                    <datef value="20010522">22th May 2001</datef>
                </date>
                <companies>
                    <company>
                        <compn>CheckPoint</compn>
                        <organizer>Carlos</organizer>
                        <compc>824357985</compc>
                    </company>
                    <company>
                        <compn>Remedy</compn>
                        <organizer>Bruno</organizer>
                        <compc>218705464</compc>
                    </company>
                </companies>
                <title>EBusiness2000 - <compn>CheckPoint</compn> and
                <compn>RemedyA</compn> and <compn>CA</compn>
                </title>
```

```
            </event>
        </events>
    </room>
</reservations>
```

The generated file:

```
<?xml version="1.0" encoding="iso-8859-1" standalone="yes"?>
<doc-status>
    <err-message>
            The final date: 22th May 2001 occurs before the beginning date:
            24th May 2001 -this is not allowed.
        </err-message>
    <err-message>
            The contact for the company CheckPoint is not a valid phone number.
        </err-message>
    <err-message>
        The title of the event must not contain any company's name outside the set
        of organizer companies, as RemedyA in a reservation of the room
        s3.
      </err-message>
    <err-message>
        The title of the event must not contain any company's name outside the set
        of organizer companies, as CA in a reservation of the room
        s3.
      </err-message>
</doc-status>
```

Application 7:

Documents where simultaneously, the value of one or more floor attributes descendant of the room elements is greater than 12; for one or more events, the final date occurs before the beginning date; one or more contacts have an invalid format; and one or more compn elements occuring inside a certain title element refer to a non-organizer company. For instance if we have one room indicated to be in the 14th floor, where occurs only one event with the final date before the beginning date, one contact with a not allowed form and one compn element occuring in the title being of a non-organizer company, the result will be a document showing the errors.

XML instance:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE reservations SYSTEM "reserv.dtd">
<reservations>
    <room floor="17" id="s3">
        <gicharge>
            <picharge>José</picharge>
            <picharge>Rita</picharge>
        </gicharge>
        <events>
            <event>
                <date>
                    <dateb value="20010524">24th May 2001</dateb>
                    <datef value="20010522">22nd May 2001</datef>
                </date>
                <companies>
                    <company>
                        <compn>CheckPoint</compn>
                        <organizer>Esteves</organizer>
                        <compc>824357985</compc>
                    </company>
                </companies>
                <title>EBusiness2000 - <compn>CheckPoint</compn> and
                <compn>CA</compn>
                </title>
```

```
            </event>
        </events>
    </room>
</reservations>
```

The generated file:

```
<?xml version="1.0" encoding="iso-8859-1" standalone="yes"?>
<doc-status>
    <err-message>
            The floor number 17 does not exist.
        </err-message>
    <err-message>
            The final date: 22nd May 2001 occurs before the beginning date:
            24th May 2001 -this is not allowed.
        </err-message>
    <err-message>
            The contact for the company CheckPoint is not a valid phone number.
        </err-message>
    <err-message>
        The title of the event must not contain any company's name outside the
        set of organizer companies, as CA in a reservation of the room
        s3.
      </err-message>
</doc-status>
```

Conclusion

For now, we can say that XCSL has proofed to be useful. It was used in many case studies with success; it means that we were able to describe the necessary constraints in a simple way, and the semantic validation processed the documents properly [4].

In this tutorial we have shown only one case-study, once its constraints address all possible categories, being, therefore, good to show several approaches to semantic problems. Other case-studies may be found in [4].

XCSL is not intended to be used as a standalone validating application but together with DTDs or XML-Schemas. However, there are situations for which we do not need a XML-Schema, once we just want to validate a particular invariant or context condition. In these cases we can and should use XCSL as a standalone application. Some reasons to use XCSL are:

- XCSL is XML. So we can use all the available tools to manipulate and process XCSL specifications.
- XML-Schema or DTDs are not always necessary, sometimes a simple XCSL specification can solve the problem.

The main ideas coming from this work can be summarized as:

- Do it simple
- Do it with existing technology

In the future, the development of XCSL will continue in three main lines:

1. The generation of Perl (XML::DT) instead of XSL. This would allow to have Perl actions and to use Perl operators in context conditions making the language and the whole system more powerful.
2. The reverse engineering of the whole system: trying to find a suitable abstract representation for these constraints and the whole model (probably High Order Attribute Grammars).
3. To upgrade the whole system, the language and the processor, to XSL 2.0 including new features like the use of quantifiers inside constraints.

References

1 Ramalho, J.C. *Anotação Estrutural de Documentos e sua Semântica.* Universidade do Minho - Portugal. 2000.

2 Ramalho, J.C., and Henriques, P.R. Constraining Content: Specification and Processing. In: *Proceedings of XML Europe'2001* , Internationales Congress Centrum (ICC), Berlin, Germany, 2001.

3 Ramalho, J.C., and Almeida, J.J. XML::DT - a Perl Down Translation Module. In *Proceedings of XML Europe'99*, Granada - Espanha, 1999.

4 Jacinto, M.H., Librelotto, G.R., Ramalho, J.C., and Henriques, P.R. Constraint Specification Languages: comparing XCSL, Schematron and XML-Schemas. In: *Proceedings of XML Europe'2002* , Barcelona, Spain, 2002.

5 Robie, J., Lapp, J., and Sach, D. *XSL Transformations (XSLT) - version 1.0.* In http://www.w3.org/TR/1998/WD-xsl-19980818, 2000.

Biography

With a degree on Applied Mathematics and Computation, *Marta Jacinto* is currently working for ITIJ - the Computer Department of the Ministry of Justice as Systems Engineer. As a researcher, she is working on her Master thesis under the subject "Semantic Validation of XML Documents".

Giovani Librelotto has a degree and a Master on Computer Science. He is currently researching on XML and Topic Maps and is preparing his Ph.D. thesis.

J. C. Ramalho is an Auxiliary Professor at the Computer Science Department of the University of Minho. He has a Masters on "Compiler Construction" and a Ph.D. on the subject "Document Semantics and Processing". He has been managing several XML projects and consulting.

Pedro Henriques is an Associated Professor of Computer Science at University of Minho. His research and teaching activity has been concerned with programming in general - paradigms, specification formalisms and languages; in particular, his main interest is the development of language processors. He completed, some years ago, his Ph.D. at University of Minho in the area of Attribute Grammars; he is, now, the leader of the "Language Specification and Processing" group. The application of the "grammatical approach to problem solving" and the use of "parsing and semantic analysis technologies" in various problem domains (namely, document processing, information retrieval and data/text mining, and geographical information systems) are the present concerns of his academic work. and xsl:value-of